```python
from datetime import datetime, timedelta
from dateutil.parser import parse
import random
import pandas as pd
import numpy as np
import sklearn
from sklearn import cluster
from sklearn.metrics import silhouette_score
from sklearn import preprocessing
import matplotlib.pyplot as plt
from matplotlib import cm

class CreatingDB:
    """
    Class for creating random database
    """
    num_people = 0  # number of people to create
    base_date = None  # the base date of data

    def __init__(self, num_people, base_date):
        self.num_people = num_people
        self.base_date = base_date

    def generate_incurred_date(self):
        """
        function to create random incurred date
        :return:
            incurred_date: string, the day of infection or contact
            elapsed_days: int, the difference between base date and incurred date
        """
        elapsed_days = random.randint(0, 14)  # the valid day period is 0~14
        # extracting the incurred day using periods and base date
        incurred_date = (self.base_date - timedelta(days=elapsed_days)). \
            strftime("%Y %m %d")
        return incurred_date, elapsed_days

    def generate_address_list(self):
        """
        function to get one address randomly from the adress list
        :return: the randomly generated address list
        """
        with open('./Address_Part.txt', 'r', encoding='utf-8') as add_file:
            # add_file = add_file.encoding
            address_list = add_file.readlines()

            random_address_list = []  # list to store addresses

            # extract addresses as many as the number of recipients
            for _ in range(1, self.num_people + 1):
                random_address_list.append(random.choice(address_list))

        return random_address_list

    def generate_status_list(self):
        # entire people num = healthy + contacted + confirmed
        num_healthy = round(self.num_people / 3)  # 1/3 is healthy
        num_contacted = round(self.num_people / 3)  # 1/3 is contacted
        # 1/3 is confirmed
        num_confirmed = self.num_people - num_healthy - num_contacted

        # add the number of states to the list
        status_list = ['Healthy' for _ in range(num_healthy)]
        status_list += ['Contacted' for _ in range(num_contacted)]
        status_list += ['Confirmed' for _ in range(num_confirmed)]
        # print(status_list)

        # shuffle for randomness
```

```
68          random.shuffle(status_list)
69
70          return status_list
71
72      def generate_csv_data(self):
73          with open('./Address_Part.txt', 'r', encoding='utf-8') as add_file:
74              # add_file = add_file.encoding
75              address_list = add_file.readlines()
76
77          status_list = self.generate_status_list()
78          print(status_list)
79          print(len(status_list))
80
81          df = pd.DataFrame(columns=("ID", "Age", "Address", "Covid Status", "Incurred
            Date"))
82          for idx in range(100):
83              row = [idx+1,
84                     random.randint(1, 100),
85                     random.choice(address_list)[:-1]]
86              print(len(status_list))
87              status = status_list.pop()
88              if status == 'Healthy':
89                  row.append(status)
90                  row.append(None)
91              else:  # Contacted or Confirmed
92                  row.append(status)
93                  date, _ = self.generate_incurred_date()
94                  row.append(date)
95              df.loc[idx] = row
96          df.to_csv("corona_data.csv", mode='w', encoding='utf-8-sig')
97
98      def old_generate_csv_data(self):
99          """
100         function to create .csv file with randomly generated records
101         :return: None
102         """
103         num_healthy = round(self.num_people / 3)  # 1/3 is healthy
104         num_contacted = round(self.num_people / 3)  # 1/3 is contacted
105         # 1/3 is confirmed
106         num_confirmed = self.num_people - num_healthy - num_contacted
107
108         id_list = list(range(1, self.num_people + 1))  # ID as many as people
109         random.shuffle(id_list)  # shuffle list
110
111         # age records as many as people
112         age_list = list(random.randint(1, 100)
113                         for _ in range(1, self.num_people + 1))
114         # address records as many as people
115         address_list = self.generate_address_list()
116
117         severity_list = []  # severity records as many as people
118         incurred_date_list = []  # incurred date list including 'None'(healthy)
119         status_list = []  # status(Healthy, Contacted, and Confirmed) list
120
121         # Entire people num = healthy + contacted + confirmed
122         # Repeat as many healthy people
123         for _ in range(num_healthy):
124             # severity_list.append(0)
125             status_list.append('Healthy')
126             incurred_date_list.append('None')
127
128         # Repeat as many contacted people
129         for count in range(num_contacted):
130             date, days = self.generate_incurred_date()
131             status_list.append('Contacted')
132             # severity_list.append(round(self.compute_severity('contacted', days), 2))
133             incurred_date_list.append(date)
```

```
134
135         # Repeat as many confirmed people
136         for _ in range(num_confirmed):
137             date, days = self.generate_incurred_date()
138             status_list.append('Confirmed')
139             # severity_list.append(round(self.compute_severity('confirmed', days), 2))
140             incurred_date_list.append(date)
141
142         # converting as pandas DataFrame data type to save .csv
143         df = pd.DataFrame({
144             "ID": id_list,
145             "Age": age_list,
146             "Address": address_list,
147             "Covid Status": status_list,
148             # "Severity": severity_list,
149             "Incurred Date": incurred_date_list,
150         })
151         df = df.sort_values(['ID'], ascending=[True])
152         df.reset_index(drop=True, inplace=True)
153
154         # saving as .csv file
155         df.to_csv("corona_data.csv", mode='w', encoding='utf-8-sig')
156
157
158 if __name__ == '__main__':
159     # CODE FOR CREATING DATABASE
160     # require the number of people and base date
161     num_people = int(input("Enter the number of people: "))
162     date_input = input("Enter the base date(Year-Month-Day): ")
163     if date_input == '':
164         print("The base date is set as today.")
165         date = datetime.now().date()
166     else:
167         date = parse(date_input).date()
168
169     cdb = CreatingDB(num_people, date)  # creating instance
170     cdb = cdb.generate_csv_data()
171     # cdb.old_generate_csv_data()  # creating .csv file
172
173
```